CENTRAL FAX CENTER

DEC 18 2006

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 100200402-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Samo Zorc

Application No.: 10/086,939

Filing Date: 02/28/2002

Confirmation No.: 4431

Examiner: Qamrun Nahar

Group Art Unit: 2191

Title: METHOD AND SYSTEM FOR AUTOMATICALLY GENERATING SOURCE CODE BASED ON A MARK-UP LANGUAGE MESSAGE DEFINITION

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on October 17, 2006.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) $500.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

☐ (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d)) for the total number of months checked below:

☐ 1st Month $120 ☐ 2nd Month $450 ☐ 3rd Month $1020 ☐ 4th Month $1590

☐ The extension fee has already been filed in this application.

☒ (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account 08-2025 the sum of $ 500. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees.

☒ A duplicate copy of this transmittal letter is enclosed.

☐ I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450
Date of Deposit:

OR

☒ I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (571)273-8300.
Date of facsimile: December 18, 2006
Typed Name: Kalyn Black
Signature: Kalyn Black

Respectfully submitted,
Samo Zorc
By RWNelson

Robert W. Nelson, Esq.
Attorney/Agent for Applicant(s)
Reg No.: 37,898
Date: December 18, 2006
Telephone: (303) 298-9888

Rev 10/06e (ApBrief)

RECEIVED
CENTRAL FAX CENTER
DEC 18 2006

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Application of:	
Samo Zorc	
Serial No.: 10/086,939	Group Art Unit: 2191
Filed: February 28, 2002	Examiner: Qamrun Nahar
For: METHOD AND SYSTEM FOR AUTOMATICALLY GENERATING SOURCE CODE BASED ON A MARK-UP LANGUAGE MESSAGE DEFINITION	Atty. Docket: 100200402-1 Confirmation No.: 4431

APPEAL BRIEF

To: Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In response to the Final Office Action mailed on July 18, 2006, and a Notice of Appeal mailed on October 17, 2006, the applicant appeals as follows:

12/19/2006 MGEBREH1 00000115 082025 10086939
01 FC:1402 500.00 DA

This brief contains items under the following headings as required by 37 CFR §41.37 and MPEP §1206:

I. Real Party In Interest
II. Related Appeals, Interferences and Judicial Proceedings
III. Status of Claims
IV. Status of Amendments
V. Summary of Claimed Subject Matter
VI. Grounds of Rejection to be Reviewed on Appeal
VII. Argument
VIII. Claims
IX. Evidence
X. Related Proceedings

Appendix A	Claims
Appendix B	Evidence
Appendix C	Related Proceedings

DEC 18 2006

(I) REAL PARTY IN INTEREST

The real party in interest in the above-referenced patent application is the Hewlett-Packard Development Company, L.P., 20555 SH 249 Houston, Texas 77070.

(II) RELATED APPEALS, INTERFERENCES AND JUDICIAL PROCEEDINGS

There are no related appeals, interferences or judicial proceedings currently known to the Appellants, Appellants' legal representatives or the assignee, which will directly affect, or be directly affected by, or have a bearing on, the Board's decision.

(III) STATUS OF CLAIMS

Claims 1, 3, 4, 6-17, and 19-30 are pending and are rejected. The rejections of all claims are appealed.

(IV) STATUS OF AMENDMENTS

No amendments were filed or entered subsequent to the final rejection mailed July 17, 2006.

(V) SUMMARY OF THE CLAIMED SUBJECT MATTER

The invention as claimed is summarized below with reference numerals and references to the specification and drawings. The invention is broadly set forth in the language corresponding to independent claims 1 and 17. Discussions about elements of the invention can be found at least in the locations in the specification and drawings cited in the claims below.

1. A method for automatically generating source code for manipulating at least one mark-up language message based on a mark-up message definition, the method comprising:

receiving the mark-up language message definition; [Fig. 3; Page 8, lines 1-5]

generating a first in-memory representation of the message definition based on the received message definition; [Fig. 3; Page 8, lines 7-10]

generating a second in-memory representation of a source code based on the first in-memory representation of the message definition, [Fig. 3; Page 8, lines 10-15] said generating a second in memory representation comprising generating a schema object tree by employing a blackboard architecture that includes agents and solutions; wherein the schema object tree includes one or more nodes; and wherein the nodes of the schema object tree are agents and the nodes of an associated source object tree are the solutions; [Page 15, lines 5-19; Page 15, lines 22-28]

generating source files based on the second in-memory representation of the source code. [Fig. 3; Page 8, lines 16-17]

17. A system for generating source code for manipulating at least one mark-up language message comprising:

a first module (114) for receiving a message definition (Fig. 2) and based thereon for generating a first in memory data structure that corresponds to the message definition, wherein the first data structure comprises a plurality of nodes; [Fig. 1; Page 6, lines 23-27; Page 8, lines 7-9

a second module (114) for receiving the first data structure and based thereon for generating a second in memory data structure that corresponds to source code for manipulating at least one mark-up language message, wherein the second data structure comprises a plurality of nodes; [Fig. 1, Page 7, lines 2-7; Page 8, lines 10-12;

a blackboard architecture, wherein the nodes of the first data structure are agents and the nodes of the second data structure are solutions. [Fig. 4; Page 15, lines 13-28]

(VI) GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1, 3, 4, 6-17, and 19-30 were rejected under 35 U.S.C. §103(a) as being unpatentable over Saulpaugh (U.S. 6,792,466) in view of Gupta (U.S. 6,513,059).

The appellant contends that the rejections are in error.

(VII) ARGUMENT

Claims 1,3, 4, 6-17, and 19-30 were rejected under 35 U.S.C. §103(a) as being unpatentable over Saulpaugh (U.S. 6,792,466) in view of Gupta (6,513,059).

CLAIM 1

Claim 1 is independent and is stated as follows for convenience:

> A method for automatically generating source code for manipulating at least one mark-up language message based on a mark-up message definition, the method comprising:
>
> receiving the mark-up language message definition;
>
> generating a first in-memory representation of the message definition based on the received message definition;
>
> generating a second in-memory representation of a source code based on the first in-memory representation of the message definition, said generating a second in memory representation comprising generating a schema object tree by employing a blackboard architecture that includes agents and solutions; wherein the schema object tree includes one or more nodes; and wherein the nodes of the schema object tree are agents and the nodes of an associated source object tree are the solutions; and
>
> generating source files based on the second in-memory representation of the source code.

According to the final office action, Saulpaugh discloses "generating a schema object tree" at column 41, lines 46-63. The applicant respectfully disagrees with the holding of the office action. The portion of Saulpaugh cited in the office action refers to a tree type display wherein services and the like are advertised as shown in Fig. 18 of Saulpaugh. The tree display is provided as a user interface and not any type of software tool as claimed in claim 1. The cited section of Saulpaugh is printed below for convenience:

> In one embodiment, a client may request a listing or tree or other representation of all services advertised in the space. The user may then scroll or maneuver through the advertisements and select the desired service. A space may also provide a look-up facility that allows a client to search for a service by providing keywords or string names. In one embodiment, a space facility may provide a mechanism to look up a space entry that has been added to the space. The look up facility may search by string to match for name, or wildcard, or even database query. The look up facility may return multiple entries from which the client may select one or perform a further narrowing search. In one embodiment, the look-up facility may provide a mechanism to locate a service advertisement matching a particular XML schema. The client may indicate a particular XML schema, or part of a particular XML, to be searched for within the space. Thus, a service may be searched for within a space according to its interface functionality.

Clearly, the tree of Saulpaugh is a display format and is not a schema object tree employing a blackboard architecture as claimed in claim 1. XML schema are matched by use of the tree display, however, the tree generated is simply the format of the display provided for a user. In other words, the tree described in Saulpaugh is the format of items displayed to a user.

The schema object tree of claim 1, on the other hand, is an operation used with a mark up language and is not a display format as described in Saulpaugh.

Examples of the schema object tree as used in the claims are provided at table 8, page 17 of the specification. Source code generated based on the schema object tree is provided at table 9, page 20 of the specification. As set forth above, the schema object tree of claim 1 is not a display as described in Saulpaugh. Just because the tree display format of Saulpaugh is a tree format used to identify XML schema does not mean that the display constitutes a schema object tree as claimed in claim 1.

In conclusion, the schema object tree of claim 1 is not a tree-type display used to advertise as disclosed by Saulpaugh. As stated above, the schema object tree of claim 1 is a programming tool or the like and not a display of advertisements.

In addition, the applicant maintains that there is no motivation for the combination of Saulpaugh and Gupta. Saulpaugh is directed toward construction of message endpoints and Gupta is directed toward a system and method for facilitating exchange of information on a computer network. Thus, the applicant contends that their combination is not proper, which renders the rejection improper.

Based on the foregoing, the cited references do not disclose all the elements of claim 1 and their combination is not proper. Therefore, the references cannot render claim 1 obvious.

The applicant respectfully requests reversal of the rejection.

CLAIM 4

Claim 4 is reprinted as follows for convenience:

> The method of claim 1 wherein the second in-memory representation includes one of class members, class methods, source file object nodes, class object nodes, and source file comment object nodes.

According to the final office action, Saulpaugh discloses a second in-memory representation including "one of class members, class methods, source file object nodes, class object nodes, and source file comment object nodes" at column 17, line 10. This section of Saulpaugh only states that some of the code may be

pregenerated for catagories (or classes) of services and then linked-in during the platform build process. This section of Saulpaugh is reprinted as follows:

> Instead, some or all of the code may be pre-generated for categories (or classes) of services, and then linked-in during the platform build process.

As stated, this section of Saulpaugh does not disclose the elements of claim 4 and cannot render claim 4 obvious. Therefore, the applicant requests that the rejection be reversed.

CLAIMS 3 AND 6-16

Claims 3 and 6-16 is dependent on claim 1 and, solely for the purposes of this appeal, will stand or fall with claim 1.

CLAIM 17

Claim 17 is independent and was rejected on the same grounds as claim 1. Claim 17 is stated as follows for convenience:

> A system for generating source code for manipulating at least one mark-up language message comprising:
>
> a first module for receiving a message definition and based thereon for generating a first in memory data structure that corresponds to the message definition, wherein the first data structure comprises a plurality of nodes;
>
> a second module for receiving the first data structure and based thereon for generating a second in memory data structure that corresponds to source code for manipulating at least one mark-up language message, wherein the second data structure comprises a plurality of nodes;

a blackboard architecture, wherein the nodes of the first data structure are agents and the nodes of the second data structure are solutions.

Claim 17 was rejected on many of the same grounds as claim 1. Therefore, the applicant incorporates the rebuttals to the rejection of claim 1 into claim 17.

As set forth above, the applicant maintains that there is no motivation for the combination of Saulpaugh and Gupta. Saulpaugh is directed toward construction of message endpoints and Gupta is directed toward a system and method for facilitating exchange of information on a computer network. Thus, the applicant contends that their combination of message endpoints and facilitating the exchange of information on a network is not proper, which renders the rejection improper.

Based on the foregoing, the applicant requests that the rejection be reversed.

CLAIMS 19-21

Claims 19-21 are dependent on claim 17 and, solely for the purposes of this appeal, will stand or fall with claim 17.

CLAIM 22

Claim 22 was rejected on the same grounds as claims 1 and 17. Therefore, the applicant applies the rebuttals to the rejection of claim 17 to this rebuttal to the rejection of claim 22.

Claim 22 is reprinted as follows for convenience:

A method for automatically generating source code for manipulating at least one mark-up language message comprising:

receiving a schema definition for a mark-up language message;

generating a first in-memory representation of the schema definition based on the schema definition;

generating a second in-memory representation of source code based on the first in-memory representation of the schema definition;

wherein the step of generating a second in-memory representation of source code based on the first in-memory representation of the schema definition includes performing one of context free processing and **context sensitive processing**.

(emphasis added)

The "context sensitive processing" element of claim 22 has not been addressed in the final office action. It was addressed in the first office action.

According to the first office action, Saulpaugh discloses "wherein the step of generating a second in-memory representation of source code based on the first in-memory representation of the schema definition includes performing one of context free processing and context sensitive processing" as claimed in claim 22. The applicant notes that the context processing is described in the present application at paragraphs 56, 62, and 77 of the printed application (page 19, lines 5-12; page 16, lines 8-16; and page 20, line 36 to page 21, line 5 of the filed application).

The first office action states that the aforementioned element of claim 22 is disclosed in Saulpaugh at column 17, line 10. This section of Saulpaugh discloses generating code, but does not disclose "performing one of context free processing and context sensitive processing" as claimed in claim 22. More specifically, this section of Saulpaugh discloses that some code need not be downloaded, which does not relate to context free or context sensitive processing. Again, there is no disclosure related to "performing one of context free processing and context sensitive processing" as claimed in claim 22.

Based on the foregoing, the applicant maintains that the rejection has been overcome and requests reversal of the rejection.

CLAIMS 23-30

Claims 23-30 are dependent on claim 22 and, solely for the purpose of this appeal, stand or fall with claim 22.

Based on the foregoing, the applicant contends that the rejections have been overcome and requests reversals of the rejections.

Respectfully submitted,
KLAAS, LAW, O'MEARA & MALKIN, P.C.

Dated: December 18, 2006

By: 

Robert W. Nelson
Reg. No. 37,898
1999 Broadway, Suite 2225
Denver, CO 80202
Tel: (303) 298-9888
Fax: (303) 297-2266

DEC 18 2006

APPENDIX A - CLAIMS

Claim 1: A method for automatically generating source code for manipulating at least one mark-up language message based on a mark-up language message definition, the method comprising:

receiving the mark-up language message definition;

generating a first in-memory representation of the message definition based on the received message definition;

generating a second in-memory representation of a source code based on the first in-memory representation of the message definition, said generating a second in memory representation comprising generating a schema object tree by employing a blackboard architecture that includes agents and solutions; wherein the schema object tree includes one or more nodes; and wherein the nodes of the schema object tree are agents and the nodes of an associated source object tree are the solutions; and

generating source files based on the second in-memory representation of the source code.

Claim 2 (cancelled)

Claim 3: The method of claim 1 wherein the first in-memory representation is a schema object tree corresponding to an XML Schema message definition; wherein the schema object tree includes one or more nodes.

Claim 4: The method of claim 1 wherein the second in-memory representation includes one of class members, class methods, source file object nodes, class object nodes, and source file comment object nodes.

Claim 5 (cancelled)

Claim 6: The method of claim 1 wherein the second in-memory representation includes elements and attributes; wherein the generating source files based on the

second in-memory representation of the source code comprises writing the elements and the attributes into respective Java class source files.

Claim 7: The method of claim 1 wherein the generating a source object tree by employing a blackboard architecture comprises performing context sensitive compilation while generating each node of the source object tree.

Claim 8: The method of claim 7 wherein the performing context sensitive compilation while generating each node of the source object tree comprises performing pre-fix processing.

Claim 9: The method of claim 7 wherein the performing context sensitive compilation while generating each node of the source object tree comprises performing in-fix processing.

Claim 10: The method of claim 7 wherein the performing context sensitive compilation while generating each node of the source object tree comprises performing post-fix processing.

Claim 11: The method of claim 1 wherein the mark-up language is XML.

Claim 12: The method of claim 1 wherein the mark-up language message definition is an XML schema message definition.

Claim 13: The method of claim 1 wherein the source code stores information included in at least one XML message.

Claim 14: The method of claim 1 wherein the source code manipulates information included in at least one XML message.

Claim 15: The method of claim 1 wherein the method generates a communication API based on an XML schema definition.

Claim 16: The method of claim 1 and further comprising automatically parsing context sensitive grammar in the compilation of XML schema to source code.

Claim 17: A system for generating source code for manipulating at least one mark-up language message comprising: a first module for receiving a message definition and based thereon for generating a first in memory data structure that corresponds to the message definition, wherein the first data structure comprises a plurality of nodes; a second module for receiving the first data structure and based thereon for generating a second in memory data structure that corresponds to source code for manipulating at least one mark-up language message, wherein the second data structure comprises a plurality of nodes; a blackboard architecture, wherein the nodes of the first data structure are agents and the nodes of the second data structure are solutions.

Claim 18 (cancelled)

Claim 19: The system of claim 17 further comprising: a mechanism for handling context sensitive grammar; wherein the processing for a current node in the first data structure considers child nodes of the current node and the parent node of the current node.

Claim 20: The system of claim 17 wherein the source code includes Java class source files.

Claim 21: The system of claim 17 wherein the mark-up language message is an XML mark-up language message.

Claim 22: A method for automatically generating source code for manipulating at least one mark-up language message comprising:

receiving a schema definition for a mark-up language message;

generating a first in-memory representation of the schema definition based on the schema definition;

generating a second in-memory representation of source code based on the first in-memory representation of the schema definition;

wherein the step of generating a second in-memory representation of source code based on the first in-memory representation of the schema definition includes performing one of context free processing and context sensitive processing.

Claim 23: The method of claim 22 further comprising generating one or more source code files based on the second in-memory representation of source code.

Claim 24: The method of claim 22 further comprising:

reading a portion of a schema definition that corresponds to one or an element or an attribute from a schema definition file;

constructing a schema object hierarchy based on the read portion;

compiling the object hierarchy into a source object hierarchy; and

writing the source object hierarchy to one or more object-oriented source files.

Claim 25: The method of claim 24 wherein schema object hierarchy includes a plurality of objects; and wherein each object includes code to compile itself into a source code primitive.

Claim 26: The method of claim 24 wherein the source object hierarchy includes a set of objects that represent a predetermined class source file and that has a predetermined number of members, methods and definitions.

Claim 27: The method of claim 24 wherein the source object hierarchy includes an object corresponding to a whole source file, an object corresponding to a file declaration comment, an object corresponding to a package name, an object corresponding to import statements, and an object corresponding to class definitions.

Claim 28: The method of claim 27 wherein the object for class definition includes one of an object corresponding to declaration statement, an object corresponding to specific class member definition, and an object corresponding to method definition.

Claim 29: The method of claim 24 wherein each source object is programmed to write itself into a respective source file.

Claim 30: The method of claim 29 wherein each source object includes a toString( ) method that recursively calls toString( ) method of its descendants to write itself into a respective source file.

APPENDIX B - EVIDENCE

There is no evidence to be presented

APPENDIX C - RELATED PROCEEDINGS

There are no related proceedings.